AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A method for dynamically adjusting the aggressiveness of an execute-ahead processor, comprising:

executing instructions in an execute-ahead mode, wherein instructions that cannot be executed because of an unresolved data dependency are deferred in a deferred buffer, and other non-deferred instructions are executed in program order, and wherein if a non-data-dependent stall condition is encountered, the execute-ahead processor enters a scout mode, wherein instructions are speculatively executed to prefetch future loads, but results are not committed to the architectural state of the execute-ahead processor;

executing deferred instructions in a deferred mode in response to determining ~~if~~ that an unresolved data dependency ~~is~~ was resolved during the execute-ahead mode;~~,~~

~~executing deferred instructions in a deferred mode;~~

~~wherein if some instructions are deferred again during the deferred mode, the method further comprises,~~

waiting for the deferred buffer to empty in response to determining ~~whether~~ that an amount of work accomplished during the execute-ahead mode exceeds a predetermined threshold; and~~,~~

~~if so,~~

~~waiting for the deferred buffer to empty, and~~

returning to a normal execution mode.~~,~~

~~otherwise resuming execution in execute ahead mode.~~

2. (Canceled).

3. (Currently amended) The method of claim 1, further comprising ~~wherein resuming execution in the non-aggressive mode involves resuming execution~~ executing instructions in a non-aggressive execute-ahead mode, wherein if a non-data-dependent stall condition is encountered, the execute-ahead processor does not enter the scout mode, but instead waits for the non-data-dependent stall condition to be resolved, or for an unresolved data dependency to return, before proceeding.

4. (Currently amended) The method of claim 1, wherein prior to executing instructions in the execute-ahead mode, the method further comprises entering the execute-ahead mode by:

issuing instructions for execution in program order during ~~a~~ the normal execution mode;

upon encountering ~~an~~ a first unresolved data dependency during execution of an instruction,

generating a checkpoint that can subsequently be used to return execution to the point of the instruction, and

executing subsequent instructions in the execute-ahead mode.

5. (Currently amended) The method of claim 4, wherein if the first unresolved data dependency is finally resolved, the method further comprises using the checkpoint to resume execution in the normal execution mode ~~from the~~

~~launch point instruction (the instruction that originally encountered the launch point stall condition).~~

6. (Original) The method of claim 1, wherein executing deferred instructions in the deferred mode involves:

issuing deferred instructions for execution in program order;

deferring execution of deferred instructions that still cannot be executed because of unresolved data dependencies; and

executing other deferred instructions that are able to be executed in program order.

7. (Currently amended) The method of claim 6, wherein if all deferred instructions are executed in the deferred mode, the method further comprises returning to ~~a~~ the normal execution mode to resume normal program execution from the point where the execute-ahead mode left off.

8. (Currently amended) The method of claim 1, wherein the unresolved data dependency ~~can include~~is one of:

a use of an operand that has not returned from a preceding load miss;

a use of an operand that has not returned from a preceding translation lookaside buffer (TLB) miss;

a use of an operand that has not returned from a preceding full or partial read-after-write (RAW) from store buffer operation; and

a use of an operand that depends on another operand that is subject to an unresolved data dependency.

9. (Currently amended) The method of claim 1, wherein the non-data-dependent stall condition ~~can include~~is one of:

a memory barrier operation;

a load buffer full condition; and

a store buffer full condition.

10. (Currently amended) An apparatus that dynamically adjusts the aggressiveness of an execute-ahead processor, comprising:

an execution mechanism configured to:

execute instructions in an execute-ahead mode, wherein instructions that cannot be executed because of an unresolved data dependency are deferred in a deferred buffer, and other non-deferred instructions are executed in program order, and wherein if a non-data-dependent stall condition is encountered, the execution mechanism is configured to enter a scout mode, wherein instructions are speculatively executed to prefetch future loads, but results are not committed to the architectural state of the execute-ahead processor;

execute deferred instructions in a deferred mode in response to determining ~~wherein if~~that an unresolved data dependency ~~is~~ was resolved during the execute-ahead mode;~~, the execution mechanism is configured to execute deferred instructions in a deferred mode;~~

wait for the deferred buffer to empty in response to determining that an amount of work accomplished during the execute-ahead mode exceeds a predetermined threshold; and to

return to a normal execution mode.

~~wherein if some instructions are deferred again during the deferred mode, the execution mechanism is configured to,~~

~~determine whether an amount of work accomplished during execute-ahead mode exceeds a predetermined threshold,~~

~~if so,~~

~~waiting for the deferred buffer to empty, and~~

~~returning to normal execution mode,~~

~~otherwise to resume execution in execute ahead mode.~~

11. (Canceled).

12. (Currently amended) The apparatus of claim 10, wherein ~~while resuming execution in the non-aggressive execution mode,~~ the execution mechanism is configured to execute instructions ~~resume execution~~ in a non-aggressive execute-ahead mode, wherein if a non-data-dependent stall condition is encountered, the execution mechanism does not enter the scout mode, but instead waits for the non-data-dependent stall condition to be resolved, or for an unresolved data dependency to return, before proceeding.

13. (Currently amended) The apparatus of claim 10, wherein prior to executing instructions in the execute-ahead mode, the execution mechanism is configured to enter the execute-ahead mode by:

issuing instructions for execution in program order during ~~a~~ the normal execution mode;

upon encountering ~~an~~ a first unresolved data dependency during execution of an instruction,

generating a checkpoint that can subsequently be used to return execution at to the point of the instruction, and

executing subsequent instructions in the execute-ahead mode.

14. (Currently amended) The apparatus of claim 13, wherein if the first unresolved data dependency is finally resolved, the execution mechanism is configured to use the checkpoint to resume execution in the normal execution mode. ~~from the launch point instruction (the instruction that originally encountered the launch point stall condition).~~

15. (Original) The apparatus of claim 10, wherein while executing deferred instructions in the deferred mode, the execution mechanism is configured to:

issue deferred instructions for execution in program order;

defer execution of deferred instructions that still cannot be executed because of unresolved data dependencies; and to

execute other deferred instructions that are able to be executed in program order.

16. (Currently amended) The apparatus of claim 15, wherein if all deferred instructions are executed in the deferred mode, the execution mechanism is configured to return to ~~a~~ the normal execution mode to resume normal program execution from the point where the execute-ahead mode left off.

17. (Currently amended) The apparatus of claim 10, wherein the unresolved data dependency ~~can include~~is one of:

a use of an operand that has not returned from a preceding load miss;

a use of an operand that has not returned from a preceding translation lookaside buffer (TLB) miss;

a use of an operand that has not returned from a preceding full or partial read-after-write (RAW) from store buffer operation; and

a use of an operand that depends on another operand that is subject to an unresolved data dependency.

18. (Currently amended) The apparatus of claim 10, wherein the non-data-dependent stall condition ~~can include~~is one of:

a memory barrier operation;

a load buffer full condition; and

a store buffer full condition.

19. (Currently amended) A computer system that dynamically adjusts the aggressiveness of an execute-ahead processor, comprising:

an execute-ahead processor;

a memory;

an execution mechanism within the execute-ahead processor configured to:

execute instructions in an execute-ahead mode, wherein instructions that cannot be executed because of an unresolved data dependency are deferred in a deferred buffer, and other non-deferred instructions are executed in program order, and wherein if a non-data-dependent stall condition is encountered, the execution mechanism is configured to enter a scout mode, wherein instructions are speculatively executed to prefetch future loads, but results are not committed to the architectural state of the execute-ahead processor;

execute deferred instructions in a deferred mode in response to determining ~~wherein if~~that an unresolved data dependency ~~is~~ was resolved during the execute-ahead mode~~;, the~~

~~execution mechanism is configured to execute deferred instructions in a deferred mode;~~

wait for the deferred buffer to empty in response to determining that an amount of work accomplished during the execute-ahead mode exceeds a predetermined threshold; and to

return to a normal execution mode.

~~wherein if some instructions are deferred again during the deferred mode, the execution mechanism is configured to,~~

~~determine whether an amount of work accomplished during execute-ahead mode exceeds a predetermined threshold,~~

~~if so,~~

~~waiting for the deferred buffer to empty, and returning to normal execution mode,~~

~~otherwise to resume execution in execute-ahead mode.~~

20. (Canceled).

21. (Currently amended) The computer system of claim 19, wherein ~~while resuming execution in the non-aggressive execution mode,~~ the execution mechanism is configured to execute instructions ~~resume execution~~ in a non-aggressive execute-ahead mode, wherein if a non-data-dependent stall condition is encountered, the execution mechanism does not enter the scout mode, but instead waits for the non-data-dependent stall condition to be resolved, or for an unresolved data dependency to return, before proceeding.